shared object is the version that includes the most recent revisions that are synchronized with the shared object and made available to other authorized users.

[**0042**] For example, a user may revise a content container of a shared object that is identified as revision R**1** by adding outline element node **370** to outline node **344** (as shown in revision R**3**). Revision R**3** is stored in the shared object. Revision R**3** is also assigned a time stamp and a GUID (e.g., GUID-**3**) to uniquely identify the revised content container. Revision R**3** is an extension of revision R**1**. Thus, revision R**1** is the latest version of the shared object that the user was aware of (e.g., the locally stored version). The shared object is inspected to determine whether the latest version of the shared object is still revision R**1**. In one embodiment, the latest version of the shared object may be determined by comparing time stamps and GUIDs of different content containers. If the latest version of the shared object is associated with a more recent time stamp than revision R**1** then another user (e.g., the user who created revision R**2**) has subsequently modified the same content container.

[**0043**] If another user has modified the same content container since revision R**1** was synchronized with the shared object, any revisions that are an extension of revision R**1** (e.g., revision R**3**) may not be synchronized with the shared object until any subsequent revisions are synchronized with the shared object and any conflicting revisions are resolved and merged. For example, revision R**2** is synchronized with the shared object after revision R**1**. Thus, the latest version of the shared object includes revision R**2**. Before revision R**3** is synchronized with the shared object, revision R**3** is compared to revision R**2** to determine if any revisions conflict. The comparison is necessary because revision R**3** is an extension of revision R**1** which is no longer associated with the latest version of the shared object. Revision R**3** is determined to not conflict with revision R**2** because outline element node **370** can be added to outline node **344** without disrupting revision R**2**.

[**0044**] In one embodiment, the shared object is revised by moving a content container from one location to another within the shared object. For example, table node **340** may be moved from page node **330** to page node **335**. A determination is made that table node **340** has been moved but the new location cannot be determined. A proxy node is created at the original location of table node **340**. The proxy node is implemented at the new location of table node **340** when the new location of table node **340** is determined. If table node **340** is deleted before the new location is determined, the proxy node is discarded.

[**0045**] Different users may simultaneously edit the shared object. Usually, the users are revising different content containers of the shared object. Thus, each user's revisions may be synchronized with the shared object without further processing. A conflict may occur when two users edit the same content container of the shared object (e.g., the same table values, the same sentence). A conflict between different user revisions may result asynchronously. For example, a user may revise a locally cached version of the shared object when not connected to a server. The revisions are synchronized with the shared object when the user reconnects to the server. However, the revisions may conflict with other revisions that have already been synchronized with the shared object.

[**0046**] For example, revision R**4** is an extension of revision R**3**. Revision R**4** deletes outline element node **350** from outline node **344**. The latest version of the shared object is determined to include revision R**2**. A comparison between revision R**2** and revision R**4** identifies a conflict because outline element node **350** is present in revision R**2** but has been deleted in revision R**4**.

[**0047**] A three-way merge is performed between a master version of a content container and two divergent versions of the content container to resolve the conflicts. For example, content container R**0** (i.e., the master version), revision R**2**, and revision R**4** are merged to establish the current version of the shared object. The master version of a content container may be the version that was last synchronized with the shared object on the server. The master version includes non-conflicting revisions.

[**0048**] The conflicting content containers are reconciled and merged into the shared object by following a set of rules established by a merge algorithm. The merge algorithm determines which revisions are synchronized with the shared object. For example, different users may be ranked according to priority such that one user's revisions take precedence over all other users (i.e., primary edits). When a lower priority user attempts to revise a content container of the shared object that has already been revised by a higher priority user, the user is informed that the revisions (i.e., secondary edits) will not be synchronized the shared object. Thus, the primary edits are displayed on a master page of the shared object and any secondary edits are flagged as not being synchronized with the shared object.

[**0049**] In another example, revisions made to a shared object on a server have priority over revisions made locally on a client. The server copy of the shared object is deemed the master version because many different users have potentially accessed and revised the shared object on the server. Only one user has accessed and revised a locally stored version. Revised content containers that are not synchronized with the shared object (e.g., secondary edits) are identified as conflicting. The conflicting content containers are preserved by being stored on conflict pages associated with the corresponding master page of the shared object.

[**0050**] **FIG. 4** illustrates a master page of a shared object and an associated conflict page. Master page **400** includes non-conflicting revisions such as content containers **410**, **420**. Any unmerged conflicting revisions are identified on master page **400** by a conflict indicator. In one embodiment, the conflict indicator is drop down menu **430**. The first entry of drop down menu **430** may be the most recent conflicts generated by the user. The entry of drop down menu **430** may include the user's name and a corresponding time stamp. Another entry in drop down menu **430** may include other conflict pages that the user generated but did not reconcile. Other entries in drop down menu **430** may correspond to conflict pages generated by other users. Selecting an entry from drop down menu **430** displays the corresponding conflict page with the conflicting revisions highlighted to draw the user's attention to the revisions that were not merged in the master version of the shared object. Thus, the user may either reconcile and merge the conflicts with master page **400** or decide that the conflicts are irrelevant.

[**0051**] In another embodiment, the conflict indicator is a tab. The master page may be associated with tab **440**.